

## REMARKS/ARGUMENTS

### Regarding Amendments

In the specification, the “Cross Reference to Related Applications” section has been amended to update the status of referenced applications. These corrections are of a clerical nature and do not add "new matter".

Claims 1 and 23-54 are now pending.

No claims stand allowed.

Claims 2-22 have been cancelled, without prejudice.

Claims 1, 32, 34-36, 38, and 40-42 have been amended to further particularly point out and distinctly claim subject matter regarded as the invention. The text of claims 23-31, 33, 37, and 39 is unchanged, but their meaning is changed because they depend from amended claims. No “new matter” has been added by the amendment.

New claims 43-54 have been added by this amendment and also particularly point out and distinctly claim subject matter regarded as the invention.

### The 35 U.S.C. §103 Rejection

Claims 1 and 23-42 stand rejected under 35 U.S.C. § 103(a) as being allegedly

unpatentable over “Java Card 2.0 Programming Concepts” by Sun Microsystems, Inc.<sup>1</sup>, among which claims 1, 32, 34-36, 38, and 40-42 are independent claims. This rejection is respectfully traversed.

According to the M.P.E.P.,

To establish a *prima facie* case of obviousness, three basic criteria must be met. First there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, not in the applicant's disclosure.<sup>2</sup>

#### Claim 1

Claim 1 as amended recites:

A small footprint device comprising:  
at least one processing element configured to execute groups of one or more program modules in separate contexts, said one or more program modules *comprising zero or more sets of executable instructions and zero or more sets of data definitions, said zero or more sets of executable instructions and said zero or more data definitions grouped as object definitions, each context comprising a protected object instance space such that at least one of said object definitions is instantiated in association with a particular context;*  
a memory comprising instances of objects; and  
a context barrier for separating and isolating said contexts, said context barrier configured for controlling execution of at least one instruction of one of said zero or more sets of instructions comprised by a program module *based at least in part on whether said at least one instruction is executed for an object instance associated with a first one of said one or more separate contexts and whether said at least one instruction is requesting access to an instance of an object definition associated with a second one of said one or more separate contexts, said context barrier further configured to prevent said access if said access is unauthorized and enable said access if said access is authorized;* and

---

<sup>1</sup> Office Action dated July 31, 2003, ¶ 2.

<sup>2</sup> M.P.E.P. §2143.

a global data structure for permitting one program module to access information from another program module by bypassing said context barrier.  
(emphasis added)

The Examiner states:

As to claim 1, SUN teaches a small footprint device (Java card / smart card) comprising: at least one processing element (virtual machine / operating system process) (pg. 3, Lifetime of the Virtual Machine) configured to execute groups of program modules (applets) in separate contexts (pg. 7, "...applets are isolated from each other." Pg. 2, "Each applet is an independent entity with its own state and functionality."), objects of a program module (objects instantiated by an applet) associated with a particular context (pg. 3, "Every object on the card is owned by the applet which instantiated it. The owning applet always has full privileges to use and modify the object."); and a context barrier (applet firewall) for separating and isolating the contexts (pg. 7, "To create a secure and trusted environment, applets are isolated from each other. An applet firewall prevents one applet from accessing the contents or behavior of objects owned by other applets."), the context barrier configured to control object-oriented access of a program module (applet) executing in one context to information (object) and/or a program module (applet) executing in another context (p 2, "However, Java Card provides facilities to support more sophisticated scenarios in which multiple applets can discover each other, communicate, and share data in a limited manner, while still maintaining protection from each other in the form of a firewall between applets.")...<sup>3</sup>

JC 2.0 states:

To create a secure and trusted environment, applets are isolated from each other. An *applet firewall* prevents one applet from accessing the contents or behavior of objects owned by other applets. Every object (class instance or array) on the card is owned by the applet which instantiated it, that is, the applet which was active at the time the object was created. The owning applet always has full privileges to use and modify the object.<sup>4</sup>

Thus in JC 2.0, an *applet* owns an object by virtue of having *instantiated* the object. JC 2.0 also attributes ownership attributes to the JCRE. JC 2.0 states:

---

<sup>3</sup> Office Action ¶ 2.

<sup>4</sup> JC 2.0, § 2.7 ¶¶ 1-2. (emphasis added)

The applet firewall ensures that no other applet may use, access, or modify the contents of an object owned by another applet except as described in this section. This does not restrict another applet from having a reference to such an object, but that applet cannot invoke methods on the object or get or set its field contents. However, it is necessary to allow exceptions to this restriction. *The JCRE must be able to invoke methods on applets, applets must be able to use objects owned by the JCRE, and applets must be able to interoperate for cooperative applications such as loyalty programs.*<sup>5</sup>

In contrast to JC 2.0, embodiments of the present invention as disclosed and claimed in claim 1 do not determine ownership of an object based upon the applet that instantiated the object. Nor are non-applet ownership attributes limited to the JCRE. Embodiments of the present invention describe objects as being *owned by a context*. With this Amendment, claim 1 has been modified to make this distinction more clear. Specifically, amended claim 1 recites in part “each context comprising a protected object instance space such that at least one of said object definitions is instantiated in association with a particular context.” Amended claim 1 also specifies that the context barrier is configured for controlling execution of at least one instruction of one of zero or more sets of instructions comprised by a program module based at least in part on whether the at least one instruction is executed for an *object instance associated with a first one of said one or more separate contexts* and whether the at least one instruction is requesting access to an instance of an *object definition associated with a second one of the one or more separate contexts*. These amendments find support in the Specification at page 19, lines 18-24, JCRE 2.1 Specification, § 6.1 (Appendix to Specification), and FIGS. 5-11 and 16-18.

---

<sup>5</sup> JC 2.0, § 2.7 ¶¶ 3-4. (emphasis added)

Accordingly, it is respectfully requested that the rejection of claims based on JC 2.0 be withdrawn. In view of the foregoing, it is respectfully asserted that the claims are now in condition for allowance.

#### Dependent Claims

Claims 23-31 depend from claim 1 and thus include the limitations of claim 1. The argument set forth above is equally applicable here. The base claims being allowable, the dependent claims must also be allowable at least for the same reasons.

#### Independent Claims 32, 34-36, 38, and 40-42

Independent claims 32, 34-36, 38, and 40-42 include limitations similar to claim 1 and thus must be allowable for at least the same reasons.

#### Claim 34

Claim 34 recites in part:

creating a global data structure which may be accessed by at least two program modules; and  
using said global data structure to permit access to information across said context barrier by bypassing said context barrier.

The Examiner states:

As to claim 34, refer to claim 1 for rejection. However, claim 34 further details the creating of the global data structure. It is obvious to one of ordinary skill in the art that since the teachings of SUN have a global data structure that it is created.<sup>6</sup>

The Applicants respectfully disagree. The cited reference does not teach using the global data structure to permit access to information across the context barrier by bypassing the context barrier. The Examiner is reminded that the mere absence from a reference of an explicit requirement of a claim cannot be reasonably construed as an affirmative statement that the requirement is in the reference.<sup>7</sup>

Dependent Claims 33, 37, and 39

Claim 33 depends from claim 32. Claim 37 depends from claim 36. Claim 39 depends from claim 38. The base claims being allowable, the dependent claims must also be allowable.

New Claims 43-54

The Examiner states:

Applicant argues that in regards to all amended claims the cited art does not teach or suggest all claim limitations. Applicant states that the invention deals with object-based access and states the prior art objects as performing code based access as the difference in not teaching or suggest all of the claim limitations. However, the examiner cannot find any disclosure within the cited art that the access is code based<sup>8</sup>

The term “code-based” refers to the sandbox model, where access control is based on the class name of the executing code, not on a non-program accessible attribute of the object instance (i.e. the owning context of the object instance) that makes the access request as presently disclosed. New claims 43-54 make this distinction more clear.

---

<sup>6</sup> Office Action ¶ 2.

<sup>7</sup> *In re Evanega*, 829 F.2d 1110, 4 USPQ2d 1249 (Fed. Cir. 1987).

Claims 43, 46, 49, and 52 specify that an object instance is associated with a context by recording the name of the context in a header of the object instance, where information in the header is inaccessible to the one or more program modules (the executing code).

Claims 44, 47, 50, and 53 specify that a memory comprises object header data that comprises information associated with at least one of the instances of objects in the memory, and that controlling execution is based at least in part on the object header data.

Claims 45, 48, 51, and 54 specify that the memory is partitioned into multiple memory spaces with instances of objects allocated for storage in one of the storage spaces, and that controlling execution is based at least in part on determining the storage space allocated to an executing object instance and an accessed object instance.

In view of the foregoing, it is respectfully asserted that the claims are now in condition for allowance.

#### Request for Allowance

It is believed that this Amendment places the above-identified patent application into condition for allowance. Early favorable consideration of this Amendment is earnestly solicited.

#### Request for Entry of Amendment

---


<sup>8</sup> Office Action ¶ 3.

Entry of this Amendment will place the Application either in condition for allowance, or at least, in better form for appeal by narrowing any issues. Accordingly, entry of this Amendment is appropriate and is respectfully requested.

If, in the opinion of the Examiner, an interview would expedite the prosecution of this application, the Examiner is invited to call the undersigned attorney at the number indicated below.

Respectfully submitted,  
THELEN REID & PRIEST, LLP

Dated: December 29, 2003

  
\_\_\_\_\_  
John P. Schaub  
Reg. No. 42,125

Thelen Reid & Priest LLP  
P.O. Box 640640  
San Jose, CA 95164-0640  
Tel. (408) 292-5800  
Fax. (408) 287-8040